UniCentral Project Documentation

Project Overview

UniCentral is a web platform designed to simplify the university application process for students in South Africa. The platform aims to unify university applications, making it easier for students to search for universities, access application information, and engage with virtual campus tours. The project includes several key features, such as a university search and filter section, interactive maps, virtual campus tours, and a real-time calendar for application dates.

Features and Functionalities

1. **University Search and Filter Section**:
    - Focuses on providing detailed information about South African universities.
    - Allows users to filter universities based on criteria such as location, courses offered, and application requirements.
2. **Interactive University Map**:
    - Features an interactive map that highlights universities supported by UniCentral.
    - Users can hover or click on the map to learn more about each university.
3. **Virtual Campus Tours**:
    - This section includes embedded videos and interactive 360-degree tours to showcase key areas of campuses, such as libraries, student centers, labs, and accommodation.
    - Utilizes card-style elements for displaying embedded videos, with images of the university that are clickable to invoke the 3D virtual tour.
4. **Calendar Integration**:
    - A real-time calendar shows university application opening and closing dates.
    - Integrates seamlessly into the website, with custom styles and colors aligning with UniCentral's design standards.
    - The calendar is designed to be simple and professional, displaying only days and navigation controls without a side column for months.
5. **Blog and News Section**:
    - Features a proper blog layout with sections for recent posts, popular posts, and a sidebar for additional content.

    - Includes a timeline section for updates on universities and their application dates, as well as a separate section for top news and real-time happenings.

6. **Responsive Design and Styling**:
    - All sections are designed to be responsive and maintain high aesthetic standards across devices.
    - Utilizes UniCentral colors (#f0c300, dark gray #222, etc.) and adheres to a clean, professional style.
7. **Contact Form and Footer**:
    - The contact form is styled similarly to HubSpot's design and is positioned within the 'Contact Us' section of the site.
    - The footer is customized to align with the overall design standards of the UniCentral website.

Design Principles

- **Color Palette**: The site uses UniCentral colors, including #f0c300 (yellow) and dark gray (#222), ensuring a consistent visual identity across the platform.
- **Professional Aesthetics**: The design emphasizes a polished and professional look, with careful attention to typography, spacing, and alignment.
- **Usability and Accessibility**: Ensures ease of use for users by creating a clear navigation structure and intuitive design elements.

Technical Implementation

- **HTML, CSS, and JavaScript**: The core technologies used for developing the frontend of the website.
- **APIs**: Integrated with a free API that provides information about South African universities, including application dates, university logos, and application fee requirements.
- **Responsive Design Techniques**: Media queries and flexible grid layouts were utilized to ensure the site is fully responsive across various devices.
- **Custom JavaScript**: Used for interactivity in the calendar section, such as date navigation and event handling.

Project Challenges and Solutions

- **Design Consistency**: Ensured uniformity in design by using consistent color palettes and styles across all sections. Refined styles like the calendar and university cards to match the overall aesthetics.
- **User Experience**: Focused on creating an intuitive user experience by simplifying the navigation and providing clear, actionable information.
- **Integration with APIs**: Successfully integrated an external API to fetch real-time data about universities, ensuring the information is up-to-date and relevant.

Future Enhancements

- **Expanded University Data**: Increase the number of universities included in the platform and expand the filter options.
- **Additional Virtual Tours**: Add more interactive and immersive virtual campus tours.
- **User Account Features**: Introduce user accounts for personalized experiences, such as saving favorite universities and tracking application statuses.

Conclusion

The UniCentral project effectively addresses the needs of prospective university students by providing a centralized, user-friendly platform for managing university applications and exploring campus life. With its combination of real-time data, interactive elements, and professional design, UniCentral is positioned as a valuable tool for students navigating the application process.

1. Navigation Button Click Event

html

Copy code

```
<script>
  document.getElementById('start-journey-btn').addEventListener('click', function() {
    window.location.href = 'loginSignIn.html';
  });
</script>
```

Explanation:

- **Purpose:** This script adds an event listener to the "Join Us" button. When the button is clicked, it redirects the user to the loginSignIn.html page.
- **How it works:**
    - document.getElementById('start-journey-btn'): This selects the button element with the ID start-journey-btn.
    - .addEventListener('click', function() {...}): This attaches an event listener to the button that listens for a click event. When the button is clicked, it executes the anonymous function provided.
    - window.location.href = 'loginSignIn.html';: This line changes the current URL to loginSignIn.html, effectively navigating to that page.

2. Toggle Information Section and Overlay

html

Copy code

```
<script>
  function toggleInfo() {
    var infoSection = document.getElementById("infoSection");
    var overlay = document.querySelector(".background-overlay");

    if (infoSection.style.display === "none") {
      infoSection.style.display = "block";
      overlay.style.display = "block";
      document.body.classList.add("active-blur");
    } else {
      infoSection.style.display = "none";
      overlay.style.display = "none";
      document.body.classList.remove("active-blur");
    }
  }

  function hideInfo() {
    document.getElementById("infoSection").style.display = "none";
    document.querySelector(".background-overlay").style.display = "none";
    document.body.classList.remove("active-blur");
  }
</script>
```

Explanation:

- **Purpose:** These functions manage the visibility of the infoSection (an informational popup) and an overlay (background-overlay) that blurs the background when the popup is visible.
- **How it works:**
    - **toggleInfo() Function:**
        - This function is called when the "Get Started" button is clicked.
        - var infoSection = document.getElementById("infoSection");: Selects the info section element by its ID.
        - var overlay = document.querySelector(".background-overlay");: Selects the overlay element using a class selector.

        - The if statement checks if the info section is currently hidden (style.display === "none").
            - If hidden, it sets infoSection.style.display and overlay.style.display to "block", making both visible.
            - It also adds a class active-blur to the body to apply a blur effect.
            - If the info section is visible, it hides both the info section and the overlay and removes the blur effect.
    - **hideInfo() Function:**
        - Hides both the infoSection and the overlay by setting their display property to "none".
        - Removes the blur effect by removing the active-blur class from the body.

3. Logo Scrolling Animation

html

Copy code

```
<script>
  document.addEventListener("DOMContentLoaded", function() {
    const logoWrapper = document.querySelector('.logo-wrapper');
    const logos = document.querySelectorAll('.logos');
    let totalWidth = 0;

    // Calculate total width of logos
    logos.forEach(logoSet => {
      totalWidth += logoSet.offsetWidth;
    });

    // Duplicate the logos to create a seamless loop
    logoWrapper.innerHTML += logoWrapper.innerHTML;

    // Set CSS animation dynamically based on the total width
    logoWrapper.style.width = `${totalWidth * 2}px`;

    const styleElement = document.createElement('style');
```

```
    styleElement.innerHTML = `
      @keyframes scroll {
        0% { transform: translateX(0); }
        100% { transform: translateX(-${totalWidth}px); }
      }
    `;
    document.head.appendChild(styleElement);
  });
</script>
```

Explanation:

- **Purpose:** This script creates a seamless scrolling animation of university logos in the #university-logos section.
- **How it works:**
    - document.addEventListener("DOMContentLoaded", function() {...}): This ensures the script runs after the DOM is fully loaded, preventing errors if elements are not yet available.
    - const logoWrapper = document.querySelector('.logo-wrapper');: Selects the wrapper containing the logo elements.
    - const logos = document.querySelectorAll('.logos');: Selects all elements with the class .logos.
    - let totalWidth = 0;: Initializes a variable to store the total width of the logos.
    - **Calculate total width of logos:**
        - logos.forEach(logoSet => { totalWidth += logoSet.offsetWidth; });: Iterates over each .logos set and sums up their widths to determine the total width required for scrolling.
    - **Duplicate the logos to create a seamless loop:**
        - logoWrapper.innerHTML += logoWrapper.innerHTML;: Appends a duplicate of the logos to create an infinite scrolling effect.
    - **Set CSS animation dynamically based on the total width:**
        - logoWrapper.style.width = ${totalWidth * 2}px;: Doubles the width of the logo wrapper to accommodate the duplicated content.
        - Creates a <style> element dynamically to define a CSS @keyframes animation that scrolls the logos horizontally from left to right.
        - @keyframes scroll: Defines the scrolling animation that moves the logos from translateX(0) to translateX(-totalWidth).

4. Virtual Tour Button Event Listeners

```
html

Copy code

<script>
   document.addEventListener("DOMContentLoaded", function() {
      const tourButtons = document.querySelectorAll('.tour-button');


      tourButtons.forEach(button => {
         button.addEventListener('click', function() {
            const tourUrl = this.getAttribute('data-tour-url');
            window.open(tourUrl, '_blank');
         });
      });
   });
</script>
```

Explanation:

- **Purpose:** This script adds click event listeners to each "3D Tour" button in the Virtual Campus Tours section, allowing users to open the 3D tour in a new tab.
- **How it works:**
    - document.addEventListener("DOMContentLoaded", function() {...}): Ensures the script runs after the DOM is fully loaded.
    - const tourButtons = document.querySelectorAll('.tour-button');: Selects all buttons with the class .tour-button.
    - tourButtons.forEach(button => {...}): Loops through each button and attaches an event listener.
    - button.addEventListener('click', function() {...}): Adds a click event listener to each button.
    - const tourUrl = this.getAttribute('data-tour-url');: Gets the URL of the tour from the data-tour-url attribute of the clicked button.
    - window.open(tourUrl, '_blank');: Opens the tour URL in a new browser tab.

Summary

- **Navigation Button Click Event**: Redirects the user to another page when a button is clicked.

- **Toggle Information Section and Overlay**: Manages the display of an informational popup and an overlay to focus user attention.
- **Logo Scrolling Animation**: Creates a seamless horizontal scrolling animation for a set of logos.
- **Virtual Tour Button Event Listeners**: Opens 3D virtual tours in new tabs when buttons are clicked.